DAYANANDA SAGAR UNIVERSITY

Devarakaggalahalli, Harohalli
Kanakapura Road, Ramanagara - 562112, Karnataka, India

SCHOOL OF ENGINEERING

Bachelor of Technology in
COMPUTER SCIENCE AND ENGINEERING

FULL STACK DEVELOPMENT
Mini Project Report

Weather App

By

Guru Kumar	**ENG23CS0070.**
Basanagouda	**ENG23CS0028.**
Chinmay M R	**ENG23CS0047.**
C.A.Ananda Vardhan	**ENG23CS0039.**

Under the supervision of
Prof. Yashpal Gupta S
Assistant Professor, Department of CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING,
SCHOOL OF ENGINEERING
DAYANANDA SAGAR UNIVERSITY,

(2024-2025)

DAYANANDA SAGAR UNIVERSITY

SCHOOL OF ENGINEERING

Department of Computer Science & Engineering

Devarakaggalahalli, Harohalli
Kanakapura Road, Ramanagara - 562112, Karnataka, India

CERTIFICATE

This is to certify that the **FULL STACK DEVELPOMENT** Mini Project work titled **"Weather-App"** is carried out by **Guru Kumar (Eng23cs0070), Basanagouda (Eng23cs0028),Chinmay M R(Eng23cs0047),C.A.Ananda Vardhan(Eng23cs0039),** bonafide students of Third semester of Bachelor of Technology in Computer Science and Engineering at the School of Engineering, Dayananda Sagar University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year **2024-2025**.

Prof. Yashpal Gupta S
Assistant Professor
Dept. of CS&E,
School of Engineering
Dayananda Sagar University
Date:

Dr. Girisha G S
Chairman CSE
School of Engineering
Dayananda Sagar University

Date:

Name of the Examiner

Signature of Examiner

DECLARATION

We, **Guru Kumar (Eng23cs0070), Basanagouda (Eng23cs0028), Chinmay M R(Eng23cs0047),C.A.Ananda Vardhan(Eng23cs0039)** are students of Third semester B. Tech in **Computer Science and Engineering**, at School of Engineering, **Dayananda Sagar University**, hereby declare that the Mini Project titled **"Weather-App"** has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science and Engineering** during the academic year **2024-2025.**

Guru Kumar **Signature:**
ENG23CS0070

Basanagouda **Signature:**
ENG23CS0028

Chinmay M R **Signature:**
ENG23CS0047

C A Ananda Vardhan **Signature:**
ENG23CS0037

Place : Bangalore Date

Date:

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of FULL STACK DEVELOPMENT mini project work.

First, we take this opportunity to express our sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.

We would like to thank ***Dr. Udaya Kumar Reddy K R, Dean, School of Engineering & Technology, Dayananda Sagar University*** *for his constant encouragement and expert advice.*

It is a matter of immense pleasure to express our sincere thanks to ***Dr. Girisha G S, Department Chairman****,* ***Computer Science and Engineering****,* ***Dayananda Sagar University,*** *for providing right academic guidance that made our task possible.*

We would like to thank our guide ***Prof. Yashpal Gupta S****, Assistant* ***Professor****,* ***Dept. of Computer Science and Engineering****,* ***Dayananda Sagar University****, for sparing his valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.*

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in the mini Project work.

TABLE OF CONTENTS

Abstract

This project develops a full-stack weather application to provide users with real-time weather information based on specified areas. The application uses modern web technologies, including HTML,CSS,Java Script for the frontend. It integrates with the OpenWeatherMap API to fetch current weather conditions, forecasts, and other meteorological data. Users can input their desired location to receive accurate weather updates displayed in a user-friendly interface. The application features a responsive design, ensuring accessibility across various devices. Backend operations include secure API communication and efficient data management, while the frontend offers dynamic visual components for an engaging user experience. The project aims to deliver a reliable, scalable, and performant application that meets user needs for timely and precise weather information. This project demonstrates proficiency in full-stack development and the ability to create practical, real-world applications by integrating various technologies and tools. By offering an intuitive interface and accurate weather data, the application enhances user engagement and utility in daily life.

This pages utilizes modern web technologies such as HTML,CSS,Java Script to create responsive,feature-rich platform.

CHAPTER 1: INTRODUCTION

The **Weather App** project leverages modern web development technologies to create a responsive and interactive application that provides real-time weather information to users. The primary focus of this project is on utilizing frontend technologies and integrating external APIs to deliver accurate and up-to-date weather data. By excluding backend development, the application streamlines its operations, relying solely on client-side processing and external API calls. The objective is to create a robust and efficient weather application that provides essential weather information through a simple yet powerful interface. This project aims to enhance user experience by delivering accurate weather data without the need for complex backend infrastructure, demonstrating the potential of frontend technologies combined with external APIs in full-stack development.

By focusing on the frontend and API integration, this weather app serves as an excellent example of how client-side technologies can be harnessed to develop practical and user-centric applications. This approach not only simplifies the development process but also ensures scalability and ease of maintenance.

- **Need For Work/Reason of Selection:**

This project on **Weather App** gives real time weather information to the users and it works on the basis of API on openweatherapp it changes it background image according to the basis of the weather condition of a particular city and it also displays the Temperature in ^{0}C and fahrenhite.

1.1 Objective

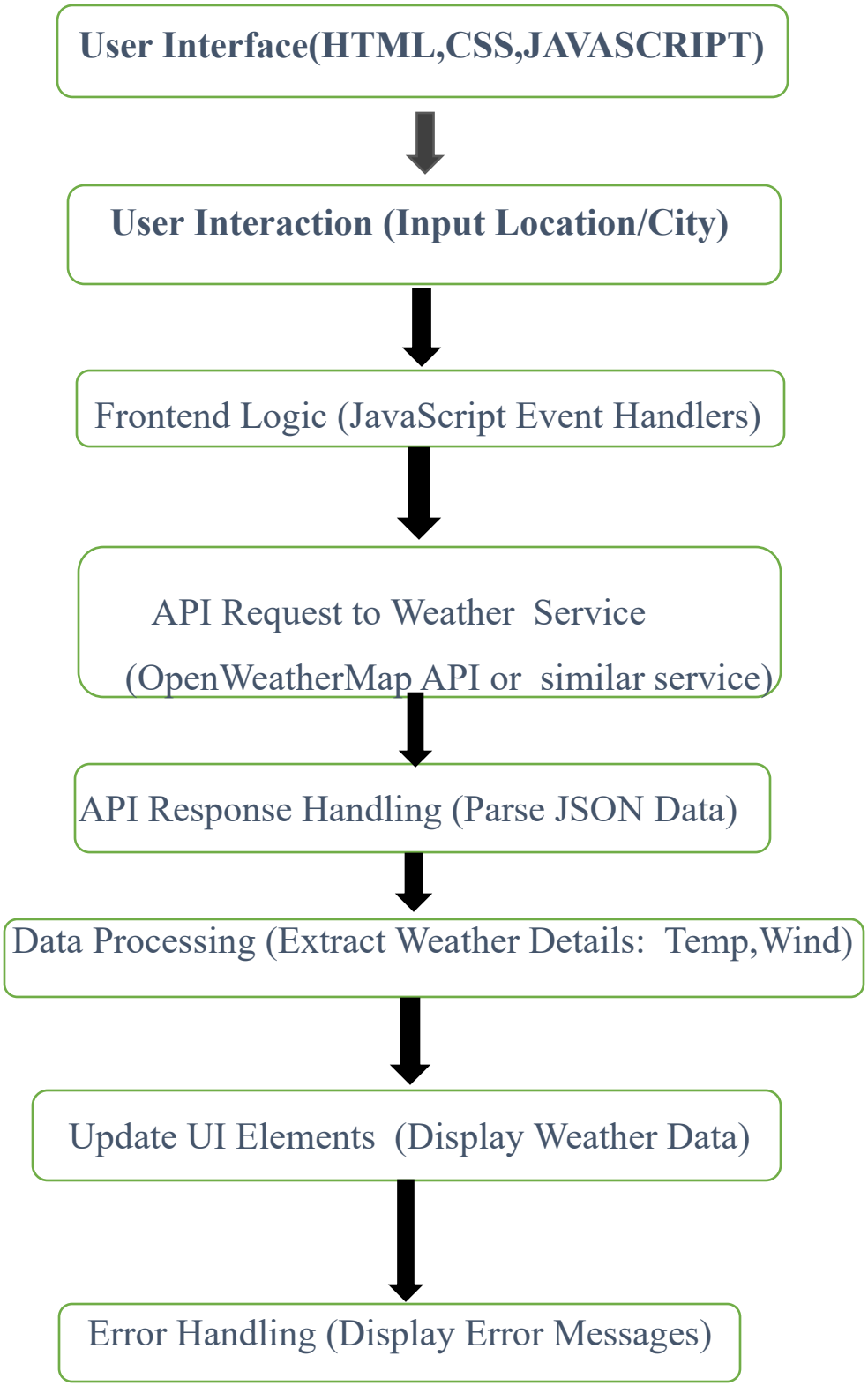
The primary objective of this mini project is to develop a weather application that delivers real-time weather information to users based on specified locations. The project aims to achieve this by leveraging modern frontend technologies and integrating external weather APIs. By focusing on client-side development, the application will provide an intuitive, responsive, and user-friendly interface that ensures quick and accurate delivery of weather data.

1.2 Scope

1. **Frontend Development**:
    - **User Interface**: Design a clean and responsive user interface using HTML, CSS, and Java Script to ensure accessibility across various devices.
    - **Dynamic Components**: Implement dynamic components to display weather data, including current conditions, forecasts, temperature, humidity, wind speed, and more.
    - **User Interaction**: Allow users to input their location or use geolocation features to automatically fetch weather information.
2. **API Integration**:
    - **Weather Data**: Integrate with external weather APIs (such as OpenWeatherMap API) to fetch real-time weather data.
    - **API Requests**: Handle API requests and responses efficiently, ensuring that the data is accurately mapped to the application's data structures.
    - **Error Handling**: Implement robust error handling to manage issues such as network errors or invalid API responses.
3. **Real-Time Updates**:
    - **Live Data**: Ensure the application updates weather information in real-time to provide users with the most current data.
    - **Responsive Design**: Develop a responsive design that adapts to various screen sizes and devices, ensuring a seamless user experience.

4. **User Experience**:
    - **Ease of Use**: Create an intuitive and easy-to-navigate interface that allows users to quickly access weather information.
    - **Visual Appeal**: Utilize modern design principles to make the application visually appealing and engaging.
5. **Performance Optimization**:
    - **Efficient Coding**: Write efficient and optimized code to ensure the application runs smoothly, even on devices with limited resources.
    - **Loading Times**: Minimize loading times by optimizing API calls and data rendering.

1.3 BLOCK DIAGRAM:



1.4 Benefits to the surrounding and the society:

1. **Skill Enhancement**:
    - **Frontend Development**: Improve your HTML, CSS, and JavaScript skills by building a practical and interactive application.
    - **API Integration**: Gain experience in integrating third-party APIs, such as the OpenWeatherMap API, to fetch and use real-time data.
2. **Practical Application**:
    - **Real-World Use**: Develop an application that provides valuable information to users, making it a useful tool for everyday life.
3. **User Interaction**:
    - **Responsive Design**: Learn to create responsive web designs that provide a seamless user experience across various devices, including desktops, tablets, and smartphones.
    - **Dynamic Content**: Enhance user interaction by updating the UI dynamically based on real-time weather data and user inputs.
4. **Time Management**:
    - **Mini Project Scope**: Work within a manageable project scope, making it easier to complete within a short timeframe while learning effectively.
    - **Efficient Coding**: Practice writing efficient and optimized code to ensure smooth performance and fast loading times.
5. **Creativity and Customization**:
    - **UI/UX Design**: Experiment with different design elements and layouts to create an appealing and user-friendly interface.
    - **Customization**: Implement features that allow users to customize their experience, such as saving preferred locations or selecting units (Celsius/Fahrenheit).
6. **Knowledge Application**:
    - **Theory to Practice**: Apply theoretical knowledge from web development courses in a practical context, reinforcing learning through hands-on experience.
    - **Real-Time Data Use**: Learn to handle and display real-time data, a crucial skill for many modern web applications.

CHAPTER-2 Methodology

➢ 2.1 Advantages:

1. **Real-Time Weather Information:**
    - Current Conditions: Provides up-to-date weather details such as temperature, precipitation, humidity, wind speed, and more.
    - Convenience: Accessible anytime and anywhere with an internet connection**.**
2. **Accessibility and Customization:**
    - User-Friendly Design: Provides easy access to weather information via desktops, tablets, and smartphones.
    - Customization: Allows users to search for specific locations, save favorite cities, or receive personalized weather updates.
3. **Unit Conversion:**
    - Provides a toggle to switch between Celsius and Fahrenheit, catering to users in different regions.
4. **. Lightweight and Fast:**
    - Uses plain JavaScript instead of heavy frameworks, ensuring quick load times and responsiveness**.**
5. **Compatibility with Modern Standards:**
    - Adheres to modern web standards, ensuring it works seamlessly across:
        - All modern browsers.
        - Various device types (desktop, tablet, and mobile).
    - Incorporates responsive design principles to optimize the app's display for different screen sizes.

➢ 2.2 Limitations:

1. **Single-Location Support**
    - Can display weather for only one location at a time, making it less useful for users who want to compare weather across multiple cities.
    - No ability to save favorite cities for quick access.

2. **Lack of Localization**

   - Does not support multiple languages or regional preferences for date/time formats or units.
   - Static content like "Loading..." and "City not found" cannot be translated for non-English-speaking users**.**

3. **Single-Location Support**

   - Can display weather for only one location at a time, making it less useful for users who want to compare weather across multiple cities.
   - No ability to save favorite cities for quick access.

➢ 2.3 Applications:

1. **Personal Weather App**
   - Allows individuals to check real-time weather information for their desired city.
   - Useful for daily planning, such as choosing appropriate clothing or scheduling outdoor activities
2. **Travel Assistance:**
   - Helps travelers plan their trips by checking the weather at their destination.
   - Can be integrated into travel platforms or standalone as a trip-planning tool.
3. **News and Media Websites:**
   - Can be embedded into news websites to provide quick weather updates alongside news articles.
   - Offers a simple, interactive weather module for users.
4. **Outdoor Activity Apps:**
   - Useful for hikers, bikers, and other outdoor enthusiasts to plan activities based on weather conditions.
   - Can include features like wind speed, or rain probability, for safety and convenience.

5. **Weather-Based Marketing:**

- Businesses can use the app to adjust marketing campaigns based on weather. For example:
  - Promoting rain gear during rainy conditions.

➢ 2.4 Purposes:

1. **Provide accurate Weather information**
   - Deliver precise and real-time weather data, including current conditions, forecasts, temperature, humidity, and wind speed, to help users plan their daily activities.
2. **Enhance user experience**
   - Create an intuitive, responsive, and visually appealing interface that allows users to easily access and understand weather information.
3. **Showcase Frontend and Api Integration skills**
   - Demonstrate the effective use of modern frontend technologies and apis in building a functional web application without relying on complex backend infrastructure.
4. **Promote learning and Skill Development**
   - Facilitate learning in full-stack development, particularly in using frontend frameworks, handling apis, and building user-centric applications.

➢ 2.5 Goals:

1. **User-Centric Design**
   - Develop a user-friendly interface with html, css, and javascript that ensures ease of navigation and quick access to weather data.
2. **Responsive Layout**
   - Ensure the application is fully responsive, providing a seamless experience across various devices, including desktops, tablets, and smartphones.
3. **Real-Time Data Integration**
   - Integrate with reliable weather apis (such as openweathermap) to fetch and display real-time weather updates accurately and efficiently.
4. **Interactive Features**
   - Implement features that allow users to input their location or use geolocation for automatic weather data retrieval, enhancing interactivity.

5. **Performance Optimization**
    - Optimize the application for fast loading times and smooth performance, ensuring a positive user experience.
6. **Error handling and Robustness**
    - Implement robust error handling mechanisms to manage api errors, invalid inputs, and network issues gracefully.
7. **Scalability and Maintainability**
    - Design the application architecture to be scalable and maintainable, allowing for future enhancements and easy updates.

➢ 2.5 Methods Used:

- **Frontend**: To build a dynamic and responsive user interface,HTML5 and CSS3 are used for structuring and styling the content, while JavaScript handles the interactive elements and API integration.

- **API**: The OpenWeatherMap API (or similar weather data providers) is used to fetch real-time weather data, including current conditions, forecasts, temperature, humidity, and wind speed.

➢ 2.6 Key Features:

- **Location-Based Weather**: Users can enter their location or allow the app to access their geolocation to get precise weather updates.
- **Real-Time Data**: The application displays real-time weather information by making API requests and updating the UI dynamically.
- **User-Friendly Interface**: The design ensures accessibility and ease of use across various devices, offering a seamless experience for users seeking quick weather updates.

CHAPTER-3 Code Snippet

➢ Html

index.html > html > head

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Weather App</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>Weather App</h1>
        <div class="weather-search">
            <input type="text" id="cityInput" placeholder="Enter city name">
            <button id="searchBtn">🔍</button>

        </div>
        <div id="loading" class="hidden">Loading...</div>
        <div id="error" class="hidden">
            <span class="error-icon">⚠️</span>
            <span id="errorMessage">City not found. Please try again!</span>
        </div>
        <div id="weatherResult" class="hidden">
            <h2 id="cityName"></h2>
            <div id="weatherIcon"></div>
            <p id="temperature"></p>
            <p id="description"></p>
            <p id="windSpeed"></p>

            <button id="unitToggle">Switch to °F</button>
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>

```

➢ CSS

style.css › #unitToggle

```css

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    background-size: cover;
    background-position: center;
    background-image: url('src/timeless-beauty-of-mount-fuji-japan-xu-1600x900 copy.jpg');
    transition: background-image 0.5s ease-in-out;
    color: #fff;
}

.container {
    background: rgba(0, 0, 0, 0.5);
    border-radius: 12px;
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.5);
    padding: 20px;
    width: 350px;
    text-align: center;
}

h1 {
    margin-bottom: 20px;
}


.weather-search {
    display: flex;
    justify-content: center;
    gap: 10px;
}

```

```css
.weather-search input {
    flex: 1;
    padding: 8px;
    border: none;
    border-radius: 8px;
    outline: none;
}

.weather-search button {
    background-color: #0072ff;
    color: #fff;
    border: none;
    border-radius: 8px;
    padding: 8px 10px;
    cursor: pointer;
    font-size: 16px;
}

.weather-search button:hover {
    background-color: #005ecb;
}


.hidden {
    display: none;
}

#error {
    margin: 10px 0;
    color: #ff4d4d;
    font-size: 14px;
}

.error-icon {
    font-size: 18px;
    margin-right: 5px;
}
```

```css
#weatherResult h2 {
    margin: 10px 0;
    font-size: 24px;
}

#weatherIcon {
    font-size: 50px;
    margin: 10px 0;
}

#temperature {
    font-size: 20px;
}

#unitToggle {
    margin-top: 10px;
    background-color: #f39c12;
    color: #fff;
    border: none;
    border-radius: 8px;
    padding: 8px;
    cursor: pointer;
}

#unitToggle:hover {
    background-color: #e67e22;
}
```

➢ JavaScript

script.js > ...

```javascript
const searchBtn = document.getElementById("searchBtn");
const cityInput = document.getElementById("cityInput");
const weatherResult = document.getElementById("weatherResult");
const loading = document.getElementById("loading");
const error = document.getElementById("error");
const cityName = document.getElementById("cityName");
const weatherIcon = document.getElementById("weatherIcon");
const temperature = document.getElementById("temperature");
const description = document.getElementById("description");
const unitToggle = document.getElementById("unitToggle");
const windSpeed= document.getElementById("windSpeed");

let isCelsius = true;

// Weather condition backgrounds
const weatherBackgrounds = {
    Clear: "url('src/sun-3588618_1920.jpg')",
    Clouds: "url('src/clouds-summer-weather-5k-1b-1600x900.jpg')",
    Rain: "url('src/rain_trees_lights_75431_1600x900.jpg')",
    Drizzle: "url('src/women-bokeh-rain-7q-1600x900.jpg')",
    Thunderstorm: "url('src/thunderstorm-01-1600x900.jpg')",
    Snow: "url('src/winter_trees_snow_133567_1600x900.jpg')",
    Mist: "url('src/car-light-road-autumn-trees-foggy-weather-y9-1600x900.jpg')",
    Haze: "url('src/fog_alone_bw_126054_1600x900.jpg')",
    Fog : "url('src/fog_alone_bw_126054_1600x900.jpg')",
    Smoke:"url('src/fog_alone_bw_126054_1600x900.jpg')",
    Default: "url('src/timeless-beauty-of-mount-fuji-japan-xu-1600x900.jpg')",
};

// Fetching weather data by city name
async function fetchWeather(city) {
    try {
        showLoading();
        hideError();
        const apiKey = "4d7fef95a3424e5c9f59307b618150ea";
        const unit = isCelsius ? "metric" : "imperial";
        const response = await fetch(
            `https://api.openweathermap.org/data/2.5/weather?q=${city}&units=${unit}&appid=${apiKey}`
        );
```

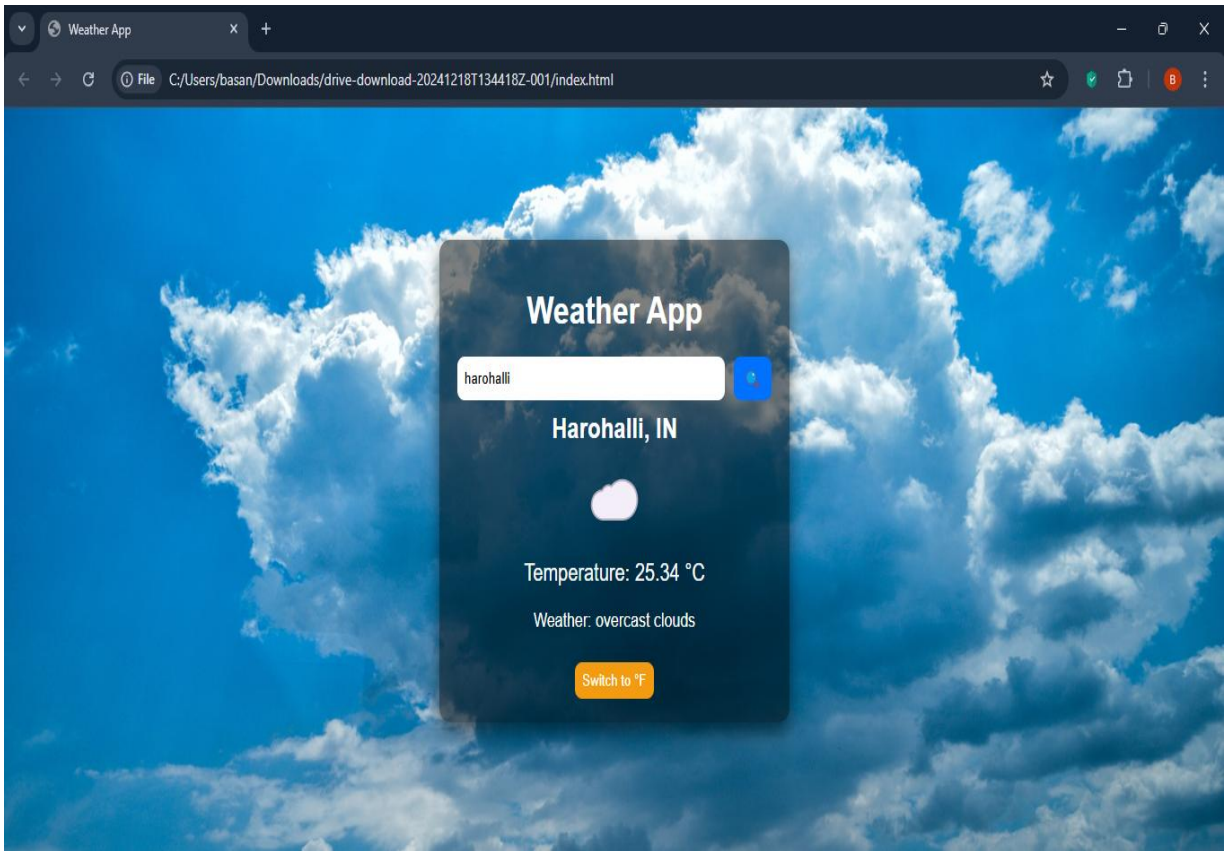
```
        if (!response.ok) throw new Error("City not found");

        const data = await response.json();
        displayWeather(data);
    } catch (err) {
        showError(err.message);
    } finally {
        hideLoading();
    }
}



// Displaying weather data
function displayWeather(data) {
    weatherResult.classList.remove("hidden");
    cityName.textContent = `${data.name}, ${data.sys.country}`;
    temperature.textContent = `Temperature: ${data.main.temp} °${isCelsius ? "C" : "F"}`;
    description.textContent = `Weather: ${data.weather[0].description}`;
    windSpeed.textContent=`Wind Speed: ${data.wind.speed} ${isCelsius ? "Km/h" : "miles/h"}`;
    weatherIcon.textContent = getWeatherIcon(data.weather[0].main);
    changeBackground(data.weather[0].main);
}

// Changing background based on condition
function changeBackground(condition) {
    const backgroundImage = weatherBackgrounds[condition] || weatherBackgrounds["Default"];
    document.body.style.backgroundImage = backgroundImage;
}

```

```
71  //  weather icon
72  function getWeatherIcon(condition) {
73      const icons = {
74          Clear: "☀️",
75          Clouds: "☁️",
76          Rain: "🌧️",
77          Drizzle: "🌦️",
78          Thunderstorm: "⛈️",
79          Snow: "❄️",
80          Mist: "🌫️",
81          Haze :"🌁",
82          Fog:"🌫️",
83      };
84      return icons[condition] || "🌍";
85  }
86
87
88  function showLoading() {
89      loading.classList.remove("hidden");
90  }
91
92  function hideLoading() {
93      loading.classList.add("hidden");
94  }
95
96  function showError(message) {
97      error.classList.remove("hidden");
98      document.getElementById("errorMessage").textContent = message;
99      weatherResult.classList.add("hidden");
100 }
101
102 function hideError() {
103     error.classList.add("hidden");
104 }
105
```

```
106
107 searchBtn.addEventListener("click", () => {
108     const city = cityInput.value.trim();
109     if (city) fetchWeather(city);
110 });
111
112
113
114 unitToggle.addEventListener("click", () => {
115     isCelsius = !isCelsius;
116     unitToggle.textContent = `Switch to °${isCelsius ? "F" : "C"}`;
117     if (cityName.textContent) fetchWeather(cityName.textContent.split(",")[0]);
118 });
119
```

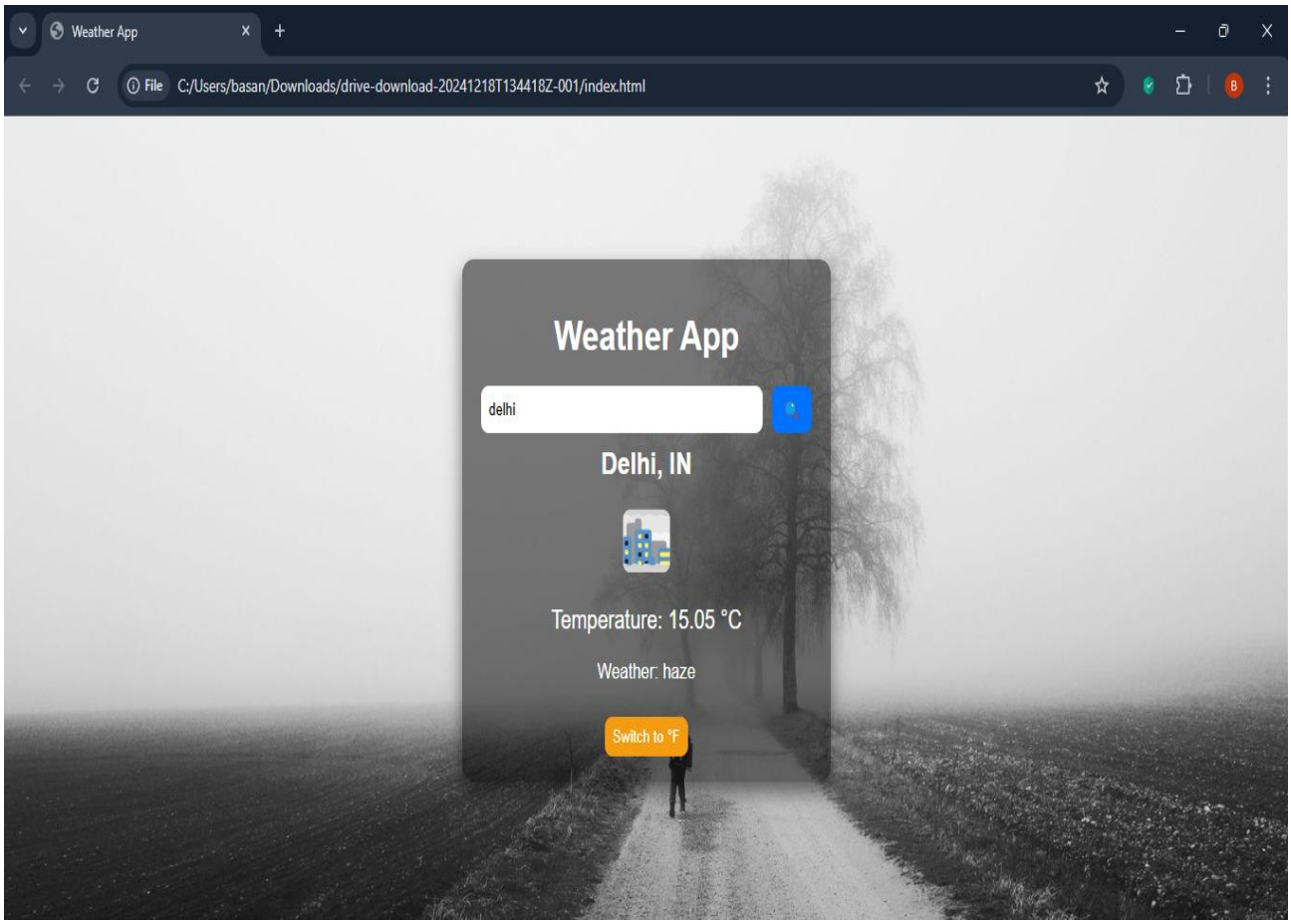
CHAPTER-4 RESULT

➢ Result 1:



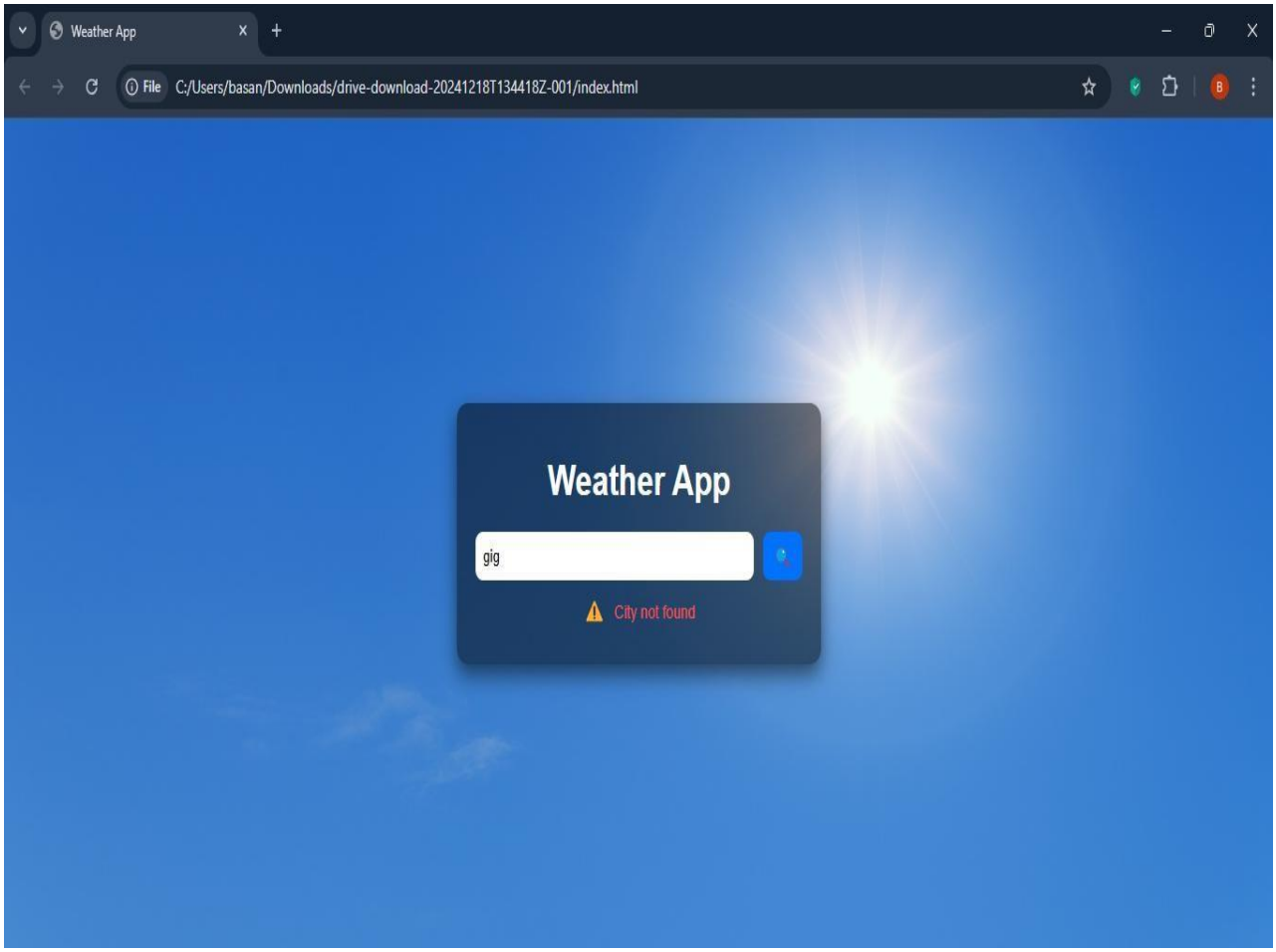
- The generated result image is the Weather report of Harohalli in Banglore. The Temperature of the city given in the result is 25.34 ^{0} C and the Weather is Overcast Clouds.

➢ Result 2:



- The generated result image is the Weather report of Delhi in INDIA. The Temperature of the city given in the result is 15.05 ^{0} C and the Weather is Haze.

➢ **Result 3:**


Weather App
File C:/Users/basan/Downloads/drive-download-20241218T134418Z-001/index.html
Weather App
gig
City not found

CHAPTER-5 Conclusion

The Weather Application project exemplifies the integration of modern frontend development techniques with powerful APIs to deliver a comprehensive and user-friendly tool for accessing real-time weather information. By leveraging HTML,CSS,JAVA SCRIPT and external weather APIs, this project successfully demonstrates the ability to create dynamic, responsive, and interactive web applications without the need for a complex backend infrastructure.

The application provides accurate weather updates, including current conditions and forecasts, in a visually appealing and easily navigable interface. With a focus on performance optimization, error handling, and user experience, the project achieves its goal of delivering a reliable and efficient weather information service.

This project serves as a testament to the potential of frontend technologies and API integration in building practical and valuable applications. It showcases the proficiency in modern web development practices and sets a solid foundation for future enhancements and scalability.